Application of AI agents for particle accelerator modeling

Y. Ivanskiy, N. Kizhaeva, A. Unzhakova

Saint Petersburg State University

25 June 2026

Why use agents at accelerators?

The practical problem

- Facilities have many control channels and subsystems.
- Knowledge is split between experts, eLogs, manuals, and archives.
- A non-routine study often needs data search, scripts, checks, and review.

What an agent can do

- Turn a request into a clear sequence of steps.
- Find supporting information in logs and documents.
- Create analysis code and save results.
- Keep machine changes limited, reviewed, and traceable.

Main idea

LLM agents should support operators and coordinate approved tools. They should not replace interlocks, fast feedback, or expert judgment.

Categorization of Papers: Topics and Coverage

Topic	Papers from the literature slide
Knowledge search	PACuna [1]; logbook RAG [2]; AccGPT [3]; AccPhysBERT [4]; document extraction [5].
Language tools	GAIA [6]; VISION [7]; Lightfall [8].
LLM tuning	LLM tuning study by Kaiser et al. [9].
Agent plans	Multi-agent proposal [10]; ALS Accelerator Assistant [11]; Osprey [12].
Long-term vision	AI-native accelerator roadmap [13].

Knowledge infrastructure: the first step

Why this comes first

- An agent needs the right context before it can plan a task.
- Useful information is spread across eLogs, manuals, chat, images, and archives.

What current work provides

- PACuna: accelerator Q&A model.
- Logbook RAG and AccGPT: search through operating knowledge.
- AccPhysBERT: better domain-specific search.
- Document extraction: keep expert knowledge available.

Safety rule

Keep the source, time, and access level with every retrieved result.

Sources: [1]–[5] on Slide 15.

Natural-language interfaces to facilities

GAIA at DESY

GAIA connects an LLM to eLogs, documentation, DOOCS information, messaging, and high-level Python scripts.

VISION and Lightfall at beamlines

These systems use small modules for command understanding, data analysis, code generation, and approved beamline APIs.

Common pattern

The LLM understands the request. Approved tools perform the action. This makes the workflow easier to check and audit.

Sources: GAIA [6]; VISION [7]; Lightfall [8].

Can an LLM tune an accelerator directly?

Test task

- Choose five magnet settings to reach target beam parameters.
- 14 LLMs and 4 prompt styles were tested.
- Results were compared with Bayesian optimization and RL-based optimization.

Main result

- LLMs improved some starting machine states.
- They were still weaker than dedicated optimizers.
- Best role today: help set up the optimization task.

Agent usage

The agent can choose the goal, variables, limits, diagnostics, and stopping rule. A trusted optimizer should then search for the best settings.

Source: Kaiser, Lauscher, and Eichler [9].

From multi-agent ideas to plan-first execution

Multi-agent idea

- Separate agents could support injection, orbit, archives, eLogs, and beamlines.
- This matches the way expertise is split across a facility.
- It is still not proven that many autonomous agents are more reliable in live operation.

What production systems do now

- Use one central coordinator.
- Build a plan before running tools.
- Show inputs, outputs, permissions, and saved results.
- Ask a human before changing hardware.

Paradigm shift

The field is moving from free-form chat to clear plans with limited, approved tool calls.

Sources: multi-agent proposal [10]; plan-first execution at ALS [11] and Osprey [12].

ALS Accelerator Assistant: multi-stage execution

Example workflow at a production synchrotron

1. Read a physics request written in normal language.
2. Find EPICS process variables and the needed time range.
3. Get past data from the archiver.
4. Create and run analysis or measurement scripts.
5. Save figures, notebooks, JSON files, and execution logs.

Result

For one measurement task, the agent prepared the work much faster than manual scripting by an expert, while following the same safety rules used by operators.

Source: Hellert et al., Accelerator Assistant at ALS [11].

Osprey: a framework for safety-critical operation

Part	What it does
Plan-first coordinator	Makes a full plan before tools are run.
Tool selector	Gives the agent only the tools needed for the current task.
Data coordinator	Moves structured data between time parsing, channel search, archive retrieval, analysis, plots, and operation.
System connectors	Connect to EPICS, archives, databases, test systems, and analysis tools.
Safe execution layer	Runs generated code in isolation and requires approval before hardware writes.

Source: Hellert, Montenegro, and Sulc [12].

Safety model for agentic execution

Read-only tasks

- Search documents, inspect logs.
- Retrieve archive data and plots.
- Run analysis code only in a sandbox
  with no write access to the control system.

State-changing tasks

- Changing setpoints.
- Enabling or disabling devices.
- Starting or stopping procedures.
- Writing configuration values.

Safety

State-changing tasks require policy checks, validated limits, and explicit operator approval before execution.

Computing requirements: what is actually needed?

Scenario	Indicative resources	Suitable tasks
Read-only pilot	16–32 CPU cores; 64–128 GB RAM; 1 TB NVMe; secure model gateway	RAG, eLog search, archive analysis, plan review, code review.
Local private service	One 24–48 GB GPU or 80 GB GPU for larger quantized models; 64–128 GB RAM	Local inference, planning, tool calls, modest concurrency.
Control-room service	A100/H100-class 80 GB GPU; 32–64 CPU cores; 128–256 GB RAM; 2 TB NVMe	Low-latency assistance, supervised multi-stage procedures, sandboxed analysis.
Resilient service	Redundant local nodes or local node plus institutional gateway	Continuity, fallback, health monitoring, safe degradation.

Published systems report Mixtral on A100-class hardware for GAIA and local H100 or institutional gateway options at ALS.

Infrastructure matters more than the GPU alone

- A local GPU is optional for an early read-only pilot if a secure model gateway is available.
- A100/H100-class local inference is useful when data must stay inside the control network, response time matters, or several users work at once.
- The agent should work on seconds-to-minutes tasks: planning, search, analysis, scripting, and supervised procedures.
- Fast feedback, interlocks, protection, and direct device control stay in EPICS/DOOCS/TANGO, PLCs, FPGAs, and approved control software.

Minimal architecture

Model endpoint + document store + control connectors + sandboxed Python workers + artifact database + monitoring + access control.

Deployment

1. **Knowledge access:** index manuals, eLogs, papers, channel catalogs, and procedures.
2. **Typed tools:** expose read-only archiver queries, PV resolution, plots, and approved analysis functions.
3. **Plan generation:** require inspectable tasks with inputs, outputs, dependencies, and access levels.
4. **Sandboxed analysis:** run generated code in isolated containers and store notebooks and figures.
5. **Bounded actuation:** allow narrow writes only with allowlists, limits, dry runs, and human approval.
6. **Continuous audit:** log prompts, plans, tool calls, artifacts, approvals, and policy violations.

Recommended pilot

Start with read-only diagnostics and shift reports. Add supervised machine procedures only after tests in a digital twin or mock control system.

Evaluation and open research questions

Evaluation metrics

- PV resolution accuracy.
- Retrieval recall and source quality.
- Whether structured plans are valid.
- Tool-call success rate.
- Time saved during preparation.
- Unsafe actions that were blocked.

Open questions

- Shared tests with realistic archives and safety policies.
- Robustness to drift, faults, and unclear machine states.
- Multimodal logs: images, waveforms, spectra, and screenshots.
- Prompt-injection and data-governance risks.
- Evidence that many agents improve reliability.

Conclusion

The near-term goal is not full autonomy. It is reliable, supervised automation that saves operator time and reduces manual scripting.

Literature

1. A. Sulc et al., *PACuna: Automated Fine-Tuning of Language Models for Particle Accelerators*, arXiv:2310.19106, 2023.
2. A. Sulc et al., *Towards Unlocking Insights from Logbooks Using AI*, arXiv:2406.12881, 2024.
3. F. Rehm et al., *AccGPT: A CERN Knowledge Retrieval Chatbot*, EPJ Web Conf. 337, 01279, 2025.
4. T. Hellert et al., *Domain-Specific Text Embedding Model for Accelerator Physics*, Phys. Rev. Accel. Beams 28, 044601, 2025.
5. Q. Dai et al., *Application of Large Language Models for the Extraction of Information from Particle Accelerator Technical Documentation*, arXiv:2509.02227, 2025.
6. F. Mayet, *GAIA: A General AI Assistant for Intelligent Accelerator Operations*, arXiv:2405.01359, 2024.
7. S. Mathur et al., *VISION: A Modular AI Assistant for Natural Human-Instrument Interaction at Scientific User Facilities*, arXiv:2412.18161, 2024.
8. R. J. Pandolfi et al., *Lightfall: An API-First, LLM-Addressable Control Platform for Synchrotron Beamlines*, arXiv:2606.06711, 2026.
9. J. Kaiser, A. Lauscher, A. Eichler, *Large Language Models for Human-Machine Collaborative Particle Accelerator Tuning through Natural Language*, Science Advances 11, eadr4173, 2025.
10. A. Sulc et al., *Towards Agentic AI on Particle Accelerators*, arXiv:2409.06336, 2025.
11. T. Hellert et al., *Agentic AI for Multi-Stage Physics Experiments at a Large-Scale User Facility Particle Accelerator*, arXiv:2509.17255, 2025.
12. T. Hellert, J. Montenegro, A. Sulc, *Osprey: Production-Ready Agentic AI for Safety-Critical Control Systems*, arXiv:2508.15066, 2025.
13. C. Tennant, *Toward a Fully Autonomous, AI-Native Particle Accelerator*, arXiv:2602.17536, 2026.